EEL 6764 **Graduate Computer Architecture** **Final Exam**
SAMPLE **Your Name:** SOLUTIONS

Notes:

- Closed Text and Closed Notes Exam
- Time: 12:30 pm – 2:30 pm
- There are **8** questions. Answer all questions. Each question carries 10 pts.
- Maximum points: 80.
- Answer in clear and legible handwriting. Partial credit will be given.

I	II	III	IV	Total
V	VI	VII	VIII	

I. (10 pts.) Instruction Set Design Principles
   a. (2 pts.) What is **Computer Architecture?**
   b. (4 pts.) Decide whether each of the following is true or false. Add brief explanation (1-2 sentences) to get full credit.
      1) The performance of the system is limited by the slowest component even if some components are made 10X faster.
      2) You can afford not to pay attention to Amdahl's law because it is not applicable anymore.
      3) MIPS rating of a processor is a good metric to measure its performance.
      4) The future of performance improvement will be mostly dependent on parallelization of programming rather than blindly adding multiple cores to a chip.
   c. (4 pts.) Some microprocessors today are designed to have adjustable voltage, so a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impact on dynamic energy and on dynamic power?

a) Computer architecture is the science and art of selecting and interconnecting hardware components to create a computer that meets functional, performance, and cost goals.

b) 1. True. Slowest portion will be the bottleneck.
2. False. Amdahl's law is still relevant, becuase the system speedup is a function of serial and parallel portions of the program/system.
3. False. You can have simple instructions in the set that can give rise to high MIPS rate for a program compared to an architecture with complex instructions. Execution time is the best metric.
4. True. If compilers can parallelize automatically then they can take better advantage of parallel cores.

c) $P = kCV^2 * f$
$P = E/T \Rightarrow E = P/f$
$P_{new} = k\,C\,(0.85\,V)^2\,(0.85)f = 0.614P$
$E_{new} = 0.614P/0.85f = 0.722E$

II. Performance Measurement

Suppose we made the following measurements:

Frequency of FP operations = 25%
Average CPI of FP operations = 4.0
Average CPI of other instructions = 1.33
Frequency of FSQRT = 2%
CPI of FSQRT = 20.

(4 pts) What is the CPI?

(6 pts) Assume that the two design alternatives are to decrease the CPI of FSQRT to 2 or to decrease the average CPI of all FP operations to 2.5. Which one is better? Provide quantitative justification.

See Page 54, Worked out example in the H&P Textbook.

a) CPI = (0.25 * 4) + (0.75 * 1.33) = 2.0

b) Optmn1 - CPI of FSQRT from 20 --> 2
CPI reduced by - (0.02 * 20) + (0.02 * 2) = -0.4 + 0.04 = -0.36
New CPI = 2 - 0.36 = 1.64
Optmn2 - average CPI of all FP from 4 -> 2.5
(0.25 * 2.5) + (0.75 * 1.33) = 1.625
Optmn 2 is slightly better.

III. **Memory Hierarchy**

a. (6 pts.) For the following advanced cache optimizations, briefly the idea.
   1) Way predicting Cache
   2) Nonblocking Cache
   3) Critical Word first and early restart

b. (4 pts.) Which is more important for floating point programs: **two-way set associativity** or **hit under one miss** for primary data caches? Assume the following average miss rates for 32 KB data caches: 5.2% for FP programs with direct mapped cache, 4.9% for programs with 2-way set associative cache. Assume the miss penalty to L2 is 10 cycles, and L2 misses and penalties are the same. Also assume hit under one miss reduces the average data cache latency for FP programs to 87.5% of a blocking cache.

III a)

1) Way Prediction - To predict the way or block inside the set of the upcoming cache access to reduce hit time, extra bits are retained in the cache. The multiplexor is set early to choose the appropriate block as a result of this prediction, and just one tag comparison is carried out that clock cycle in parallel with reading the cache data. In the event of a miss, the following clock cycle will search the other blocks for matches.

2) Nonblocking Cache - For pipelined computers that support out-of-order execution, the processor must halt on a data cache miss. By enabling the data cache to keep supplying cache hits even in the event of a miss, a nonblocking cache or lockup-free cache increases the potential benefits and increases cache bandwidth. By assisting during a miss rather than disregarding the processor's demands, this "hit under miss" optimization lowers the effective miss penalty.

3) Critical Word First/Early Restart - This method is based on the observation that the processor typically only requires one word at a time from the block.

1) Critical word first: To fill in the remaining words in the block, ask for the missing word first from memory and transmit it to the processor as soon as it is available. Then, let the processor carry on with its current task.

2) Early restart: Retrieve the words in the normal order, but as soon as the block's desired word appears, transmit it to the processor and allow it to carry out the remaining instructions.

See Example problem pages 101/102.

III (b)

The average memory stall times are:

Direct Mapped (DM): 5.2% x 10 = 0.52

2-Way: 4.9% x 10 = 0.49

Thus cache access latency (including stalls) for 2-way is 0.49/0.52 = 94% of DM cache. As hit under one miss reduced the average data cache latency for FP programs to 87.5% of a blocking cache, DM with one hit under one miss is better than the 2-way set associative cache that blocks on a miss.

IV. **Basic Pipelining**

a. (4 pts) The following series of branch outcomes occurs for a single branch in a program. (T means the branch is taken, N means the branch is not taken).

Index 1 2 3 4 5 6 7 8 9 10 11 12 13
T, T, N, T, N, T, T, T, T, N, T, T, N

Assume that we are trying to predict this sequence with a Branch History Table (BHT) using a 1-bit prediction. The counters of the BHT are initialized to the N state. Which of the branches would be mispredicted? Show their indices.

b. (6 pts) Use the following code segment:

```
Loop:ld   x1, 0(x2)  : load x1 from address 0+x2
     Addi x1, x1, 1  ; x1=x1+1
     Sd   x1, 0, (x2); store x1 at address 0+x2
     Addi x2, x2, 4  ; x2 =x2 + 4
     Sub  x4, x3, x2 ; x4 = x3 -x2
     Bnez x4, Loop   ; branch to Loop if x4!=0
```

Assume that the initial value of x3 is x2 + 396. List all of the data dependencies in the code above. Record the register, source instruction, and destination instruction.

a)
Bold font slots on third row show 8 total mis-predictions

Index	1	2	3	4	5	6	7	8	9	10	11	12	13
	T	T	N	T	N	T	T	T	T	N	T	T	N
Predictor	**N**	T	**T**	**N**	**T**	**N**	T	T	T	**T**	**N**	T	**T**

b)
x1 ld addi
x1 addi sd
x2 addi sub
x2 addi ld
x2 addi sd
x4 sub bnez

V. RISC- Architecture

Answer the following RISC-V Base ISA architecture:

1. (4 pt.) Name any *four ISA* design principles that RISC-V implements.

See Page A-33 in H&P Textbook
1) Use general-purpose registers with load-store architecture.
2) Support displacement, immediate, and register indirect addressing modes
3) Support these data sizes and types: 8-, 16-, 32-, and 64-bit integers and 64-bit IEEE FP numbers.
4) Support simple instructions that occur frequently, load, store, add, subtract, move reg-to-reg, and shift
5) Compare equal, compare not equal, compare less, branch, jump, call, and return.
6) Use fixed instruction encoding if interested in performance
7) Provide atleast 16 and preferably 32 general purpose registers

2. (1 pt.) No. of integer registers __32__
   No. of FP registers_____0__

3. (1 pt.) Instruction word length_____32_____

4. (1 pt.) No. of instruction formats _____4_____

5. (1 pt.) Value of x0 is _____0_____

6. (2 pts) Why is it called a Load-store architecture?

For any operation to be performed, first the operands must be in the registers, therefore, load and store instructions are needed to move the operands from the memory to registers and vice-versa.

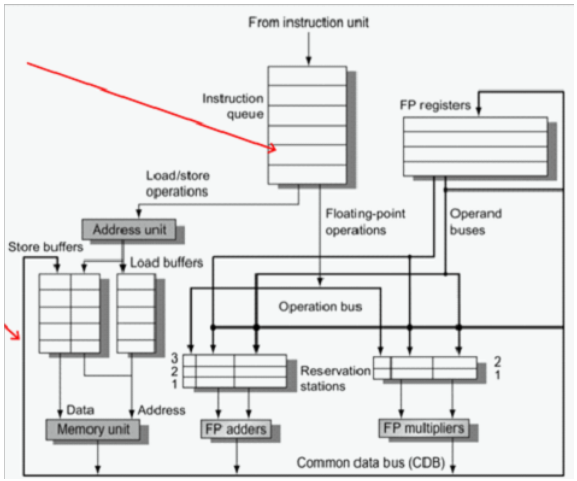
VI. **Instruction level Parallelism**

a. (6 pts) Draw the block diagram of basic structure of a RIS-V FP Unit using Tomasulo's algorithm. Briefly explain the purpose of each block. Assume FP adders, FP multipliers for arithmetic computation.

b. (4 pts) Show what the information tables look like for the following code sequence when only the first load has complete and written its result.

```
1. fld      f6, 32(x2)
2. fld      f2, 44(x3)
3. fmul.d   f0, f2, f4
4. fsub.d   f8, f2, f6
5. fdiv.d   f0, f0, f6
6. fadd.d   f6, f8, f2
```

a)



Instruction queue - contains issued instructions
Reservation stations - contains instructions that are waiting for atleast one of its operands is a result of another instruction
Common data bus - the results of an instruction executed will be broadcast on this bus which can be grabbed by waiting instructions in reservation stations, FP registers, store buffers.
Load and Store Buffers - contains data and addresses, act like reservation stations

b) See Figure 11 on Page 202 in H&P Textbook

	Instruction status		
Instruction	Issue	Execute	Write result
fld f6,32(x2)	√	√	√
fld f2,44(x3)	√	√	
fmul.d f0,f2,f4	√		
fsub.d f8,f2,f6	√		
fdiv.d f0,f0,f6	√		
fadd.d f6,f8,f2	√		

	Reservation stations						
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[x3]
Add1	Yes	SUB		Mem[32 + Regs[x2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[f4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[x2]]	Mult1		

	Register status								
Field	f0	f2	f4	f6	f8	f10	f12	...	f30
Qi	Mult1	Load2		Add2	Add1	Mult2			

VII. **Data level Parallelism**

a. (2 pts.) What is the basic idea of a vector processor?

b. (8 pts.) With an example, explain the basic idea of the following techniques:

i. Vector chaining
ii. Strip Mining
iii. Vector Stride
iv. Gather and Scatter

a) The main idea behind vector processing is performing a single operation on multiple data at the same time. So, instead of processing each data element individually, the elements are placed into vectors/ arrays in order for them to be processed in "bulk". In other words, vector processing allows computers to go from sequential data processing (one operation for each data element) to matrix or array processing (one operation for the whole array or matrix). Vector processing is also known as SIMD (Single Instruction Multiple Data).

b)

i) Vector chaining: The main idea of vector chaining is the optimization of instruction execution (vector version of forwarding). The next vector operation starts when the result from previous operation on the first element is finished.

Example:

```
LV     v1
MULV   v3, v1,  v2
ADDV   v5  v3,  v4
```

ii) Strip Mining: Strip-mining, or loop sectioning, is a technique for manipulating loops such that the loop is partitioned in smaller segments or "strips". It may improve memory performance. Reduces the loop's size by a factor of the length of the vector maximum length.

Example:

This vector loop:

```
for (i=0; i < n; i = i + 1)
  Y[i] = a * X[i] + Y[i]
```

is converted to the following:

```
low = 0;
VL = (n % MVL);
for(j=0; j <= (n/MVL); j=j+1) {
  for(i=low; i <(low + VL); i=i+1)
    Y[i] = a * X[i] + Y[i];
  low = low + VL;
  VL = MVL
}
```

iii) Vector Stride: Technique for handling multi-dimensional arrays. Stride is the distance that separates elements in an array to be gathered into a single register. Vector base address and stride are stored in GP registers.

Example:

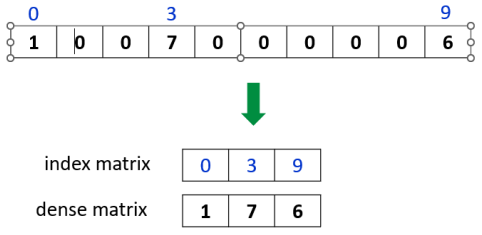
sdf

```
for(i=0; i<100; i=i+1)
  for(j=0; j<100; j=j+1) {
    A[i][j] = 0.0;
    for(k=0; k < 100; k=k+1)
      A[i][j] = A[i][j] + B[i][k]*D[k][j];
}
```

If D is stored in row-major layout then the stride is 100 double words.

iv) Gather and Scatter: A way of addressing and improving memory access in sparse matrices (arrays which most elements are zero). It basically creates a new matrix (index matrix) with the index of the elements that are non-zero and then maps it out to a new matrix (dense matrix).

Example:



VIII. **Thread Level Parallelism**

a. (5 pts.) Explain how the following protocols work:
   1) Snooping coherence protocol
   2) Directory based coherence protocol.

b. (2.5 pts.) Using a state-diagram like figure, show how a **read miss** from local node is handled by the home node and remote node. Briefly explain how it works.

c. (2.5 pts.) Using a state-diagram like figure, show how a **write miss** from local node is handled by the home node and remote node. Briefly explain how it works.

a. 1) Snooping coherence protocol

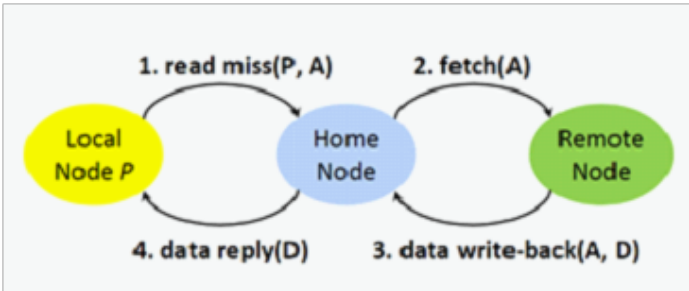
- Every cache tracks sharing status of each cache block
- Mem requests are broadcast on a bus to all cache
- Writes serialized naturally by a single bus

Used in shared memory multiprocessor systems. Each cache controller monitors, or "snoops", the bus that interconnects the processors and caches to determine if any other processors or caches have accessed the same memory block.

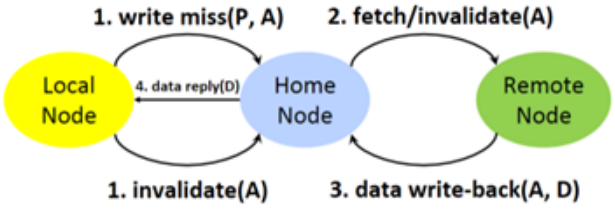
2) Directory based coherence protocol.

The memory is distributed across the processors. Directory maintains block states and sends invalidation messages, i.e., tracks states of all local memory blocks. For each memory block, shared, uncached (invalid), and modified (exclusive).

b. Read miss: When a local node has a read miss, it will request the home node for a copy (arc 1). Home node then requests the latest copy from remote node (arc 2). The remote node does the write-back to home node (arc 3). Finally, home node sends the latest copy to the requesting local node (arc 4).



b.Write miss: When a local node has a write miss, it will request the home node for a copy (arc 1) as well as invalidates the copy. Home node then requests the latest copy from remote node (arc 2) as well as invalidates the remote nodes's copy . The remote node does the write-back to home node (arc 3). Finally, home node sends the latest copy to the requesting local node (arc 4).



Operation 2: invalidate, or fetch & Invalidate

Formulas

1. $Energy \propto \frac{1}{2} \times Capacitive\ Load \times Voltage^2$
2. $Power \propto \frac{1}{2} \times Capacitive\ Load \times Voltage^2 \times Frequency\ Switched$
3. $Cost\ of\ Integrated\ Circuit = \frac{Cost\ of\ die + Cost\ of\ testing\ die + Cost\ of\ packaging\ and\ final\ test}{Final\ test\ yield}$
4. $Cost\ of\ Die = \frac{Cost\ of\ Wafer}{Dies\ per\ wafer \times Die\ Yield}$
5. $Dies\ per\ wafer = \frac{\pi \times (\frac{Wafer\ diameter}{2})^2}{Die\ Area} - \frac{\pi \times \text{Wafer Diameter}}{\sqrt{2 \times Die\ Area}}$
6. $Die\ Yield = Wafer\ Yield \times \frac{1}{(1 + Defects\ per\ unit\ area \times Die\ Area)^N}$
7. If X is 'n' times fast as Y then: $Execution\ Time(n) = \frac{Execution\ Time_Y}{Execution\ Time_X} = \frac{Performance_X}{Performance_Y}$
8. Amdahl's Law:
   $Speedup_{Overall} = \frac{Execution\ Time_{old}}{Execution\ Time_{new}} = \frac{1}{(1 - Fraction_{enchanced}) + \frac{Fraction_{enchanced}}{Speedup_{enchanced}}}$
9. $CPU\ Time = CPU\ Clock\ Cycles\ for\ a\ program \times Clock\ Cycle\ Time$
10. $CPI = \frac{CPU\ clock\ cycles\ for\ a\ program}{Instruction\ Count}$
11. $CPU\ Time = Instruction\ Count \times Cycles\ per\ Instruction \times Clock\ cycle\ Time$
12. $CPU\ Clock\ Cycles = \sum_{i=1}^{n}(IC_i \times CPI_i)$
13. $CPU\ Time = \sum_{i=1}^{n}(IC_i \times CPI_i) \times Clock\ Cycle\ Time$
14. $CPI = \frac{\sum_{i=1}^{n}(IC_i \times CPI_i)}{Instruction\ Count} = \sum_{i=1}^{n}(\frac{IC_i}{Instruction\ Count} \times CPI_i)$
15. $\frac{Misses}{Instruction} = Miss\ rate \times \frac{Memory\ accesses}{Instruction}$
16. $Average\ Memory\ Access\ Time = Hit\ Time + Miss\ Rate \times Miss\ Penalty$
17. For Multilevel Caches:
    $Average\ Memory\ Access\ Time = Hit\ Time_{L1} + Miss\ Rate_{L1} \times (Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L1})$
18. $Memory\ Stall\ Cycles = Number\ of\ Misses \times Miss\ Penalty$
    $= IC \times \frac{Misses}{Instruction} \times Miss\ Penalty$
    $= IC \times \frac{Memory\ Accesses}{Instruction} \times Miss\ Rate \times Miss\ Penalty$
19. $Memory\ Stall\ Cycles = IC \times Reads\ per\ Instruction \times Read\ Miss\ Rate \times Rate\ Miss\ Penalty + IC \times$

$$Writes\ per\ Instruction \times Write\ Miss\ Rate \times Write\ Miss\ Penalty$$

20. $$CPU\ Execution\ Time = (\ CPU\ Clock\ Cycles + Memory\ Stall\ Cycles\ ) \times Clock\ Cycle\ Time$$

21. $$CPU\ Execution\ Time = IC \times \left(CPI_{execution} + \frac{Memory\ Stall\ Clock\ Cycles}{Instruction}\right) \times Clock\ Cycle\ Time$$

22. $$CPU\ Execution\ Time = IC \times \left(CPI_{execution} + \frac{Misses}{Instruction} \times Miss\ Penalty\right) \times Clock\ Cycle\ Time$$

23. $$CPU\ Execution\ Time = IC \times \left(CPI_{execution} + Miss\ Rate \times \frac{Memory\ Accesses}{Instruction} \times Miss\ Penalty\right) \times Clock\ Cycle\ Time$$

24. $$\frac{Memory\ stall\ Cycles}{Instruction} = \frac{Misses}{Instruction} \times (\ Total\ Miss\ Latency - Overlapped\ Miss\ Latency)$$

25. $$2^{index} = \frac{Cache\ Size}{Block\ size \times Set\ Associativity}$$

26. $$\frac{Memory\ stall\ Cycles}{Instruction} = \frac{Misses_{\ L1}}{Instruction} \times Hit\ Time_{\ L2} + \frac{Misses_{\ L2}}{Instruction} \times Miss\ Penalty_{\ L2}$$

27. $$Average\ Instruction\ Execution\ Time = Clock\ Cycle \times Average\ CPI$$

28. $$Speedup\ from\ Pipelining = \frac{Average\ instruction\ time\ unpipelined}{Average\ instruction\ time\ pipelined}$$

29. $$Speedup\ from\ Pipelining = \frac{CPI\ unpipelined \times Clock\ Cycle\ Unpipelined}{CPI\ pipelined \times Clock\ Cycle\ pipelined}$$

30. $$CPI\ Pipelined = Ideal\ CPI + Pipeline\ Stall\ Clock\ Cycles\ per\ Instruction = 1 + Pipeline\ stall\ cycles\ per\ Instruction$$

31. $$Speedup = \frac{CPI\ unpipelined}{1 + Pipeline\ stall\ cycles\ per\ Instruction}$$

32. $$Speedup = \frac{Pipeline\ depth}{1 + Pipeline\ stall\ cycles\ per\ Instruction}$$

33. $$Speedup\ from\ pipelinig = \frac{1}{1 + Pipeline\ stall\ cycles\ per\ Instruction} \times \frac{Clock\ Cycle\ unpipelined}{Clock\ cycle\ pipelined}$$

34. $$Pipeline\ depth = \frac{Clock\ Cycle\ unpipelined}{Clock\ cycle\ pipelined}$$

35. $$Speedup\ from\ pipelinig = \frac{1}{1 + Pipeline\ stall\ cycles\ per\ Instruction} \times Pipeline\ depth$$

36. $$Average\ Instruction\ Time = CPI \times Clock\ Cycle\ Time$$